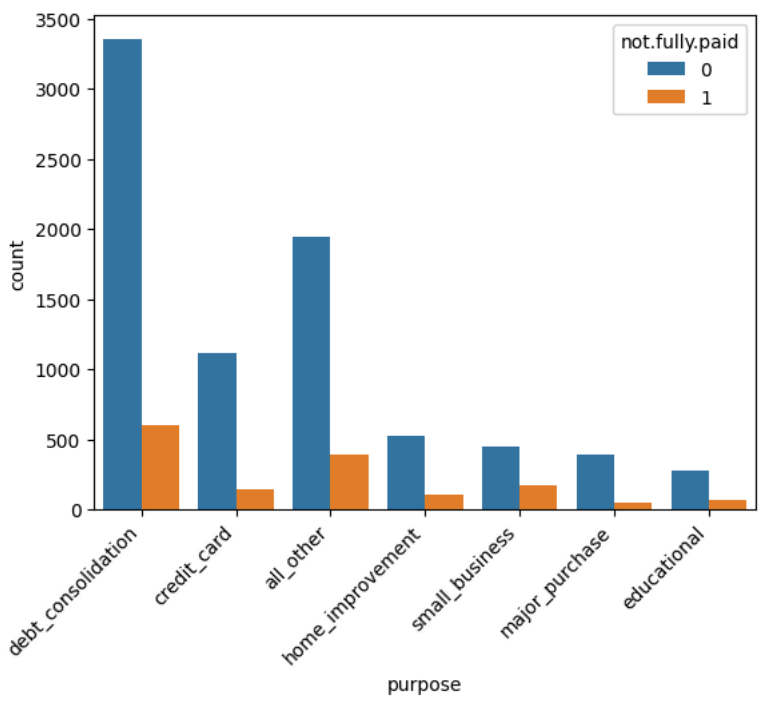
```
import pandas as pd
df = pd.read_csv("/content/loan_data.csv")
df.head()
```

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   credit.policy      9578 non-null   int64  
 1   purpose            9578 non-null   object 
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64  
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64  
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64  
 11  delinq.2yrs        9578 non-null   int64  
 12  pub.rec            9578 non-null   int64  
 13  not.fully.paid     9578 non-null   int64  
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(data=df,x='purpose',hue='not.fully.paid')
plt.xticks(rotation=45, ha='right');
```




```
pre_df = pd.get_dummies(df,columns=['purpose'],drop_first=True)
pre_df
```

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0
...	...	...	...	...	...	...	...	...	...	...
9573	0	0.1461	344.76	12.180755	10.39	672	10474.000000	215372	82.1	2
9574	0	0.1253	257.70	11.141862	0.21	722	4380.000000	184	1.1	5
9575	0	0.1071	97.81	10.596635	13.09	687	3450.041667	10036	82.9	8
9576	0	0.1600	351.58	10.819778	19.18	692	1800.000000	0	3.2	5
9577	0	0.1392	853.43	11.264464	16.28	732	4740.000000	37879	57.0	6

9578 rows × 19 columns

```
from sklearn.model_selection import train_test_split
X = pre_df.drop('not.fully.paid', axis=1)
y = pre_df['not.fully.paid']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=125)
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
```

```
GaussianNB()
```

```
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    f1_score,
    classification_report,
)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")
print("Accuracy:", accuracy)
print("F1 Score:", f1)
```

```
Accuracy: 0.8206263840556786
F1 Score: 0.8686606980013266
```

```
labels = ["Fully Paid", "Not fully Paid"]
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```